

---

## **Le condizioni di ricerca**

# Le condizioni di ricerca

- Usate nelle clausole **WHERE** e **HAVING**
- Costruite con gli operatori di confronto: **>**, **>=**, **<**, **<=**, **=**, **<>**
- Espressioni ottenute concatenando confronti con gli operatori logici: **AND**, **OR**, **NOT**, **XOR**
- Altri operatori di confronto: **BETWEEN**, **NOT BETWEEN**, **IN**, **NOT IN**, **LIKE**, **NOT LIKE**, **IS NULL** , **IS NOT NULL**
- **BETWEEN**: controlla l'appartenenza di un valore in un dato intervallo
- **IN**: controlla l'appartenenza a uno dei valori di un elenco
- **LIKE**: confronta una stringa di caratteri con un modello di stringa costruita, in genere, con **caratteri jolly**
- **IS NULL**: controlla la presenza di valori nulli

# BETWEEN, IN

- **BETWEEN**

```
SELECT Cognome, Nome, Residenza  
FROM Impiegati  
WHERE Stipendio BETWEEN 30000 AND 45000;
```

equivale a:

```
WHERE Stipendio >= 30000 AND Stipendio <= 45000;
```

- **IN**

```
SELECT *  
FROM Impiegati  
WHERE Residenza IN ('Torino', 'Venezia', 'Palermo');
```

equivale a:

```
WHERE Residenza = 'Torino' OR Residenza = 'Venezia' OR  
Residenza = 'Palermo';
```

# LIKE (1)

## Caratteri Jolly

- ? indica uno e un solo carattere qualsiasi in quella posizione della stringa (nello standard SQL \_)
- \* indica una sequenza qualsiasi di caratteri in quella posizione della stringa (nello standard SQL %)

**LIKE 'xyz\*'** → riconosce le stringhe che iniziano con 'xyz';

**LIKE '\*xyz'** → riconosce le stringhe che finiscono con 'xyz';

**LIKE '\*xyz\*'** → riconosce le stringhe di 3 o più caratteri che contengono 'xyz';

**LIKE '?xyz'** → riconosce le stringhe di 4 caratteri che finiscono con 'xyz'.

```
SELECT Cognome, Dipartimento
FROM Impiegati
WHERE Cognome LIKE 'R*';
```

Gli impiegati con il  
cognome che inizia per R

# LIKE (2)

- [ ] ricerca un carattere fra quelli elencati fra le parentesi quadre
- ! ricerca caratteri diversi da quelli specificati tra [ ] (anche ^)
- [ $\alpha$ - $\beta$ ] ricerca i caratteri di un intervallo. Deve essere  $\alpha < \beta$
- # ricerca un carattere numerico ("caratteri jolly" nella guida in linea)

**Like** "R[ai]s\*" → Raso, Riso, Rasente, Risoluto ma non Rosoni

**Like** "R[!i]s?" → Raso, Reso, Roso, ma non Rosoni, Riso, Risaia

**Like** "[A-M]\*" → tutte le stringhe che iniziano con un carattere compreso tra A ed M

```
SELECT Cognome, Dipartimento
FROM Impiegati
WHERE Cognome LIKE [Carattere iniziale?] & '*';
```

L'iniziale del cognome è scelta  
quando si esegue l'interrogazione

# Funzioni di aggregazione (1)

- **COUNT, SUM, MIN, MAX, AVG, ...**
- Sintetizzano le informazioni di una colonna in un solo valore
- Possono comparire solo nelle clausole **SELECT** e **HAVING**
- **COUNT**: conta il numero di righe o di valori non nulli in una colonna
  - **COUNT (\*)** conta le righe di una tabella
- **SUM**: somma i valori non nulli di una colonna
- **AVG**: restituisce la media dei valori non nulli di una colonna
  - **AVG (Stipendio) = SUM (Stipendio) / COUNT (Stipendio)**
- **MIN, MAX**: restituiscono il valore minimo e massimo di una colonna

```
SELECT COUNT(*) FROM Impiegati;
```

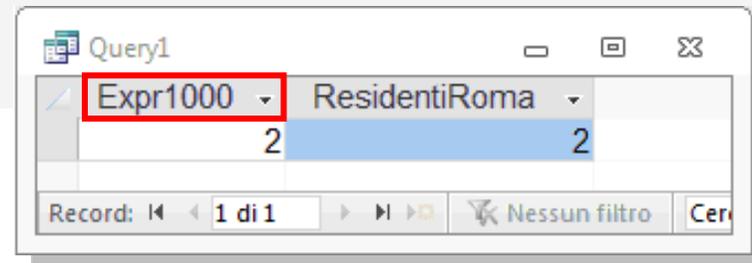
Restituisce 12

# Funzioni di aggregazione (2)

```
SELECT COUNT(Dipartimento) FROM Impiegati;
```

Restituisce 11

```
SELECT COUNT(*), COUNT(*) AS ResidentiRoma  
FROM Impiegati  
WHERE Residenza = 'Roma';
```

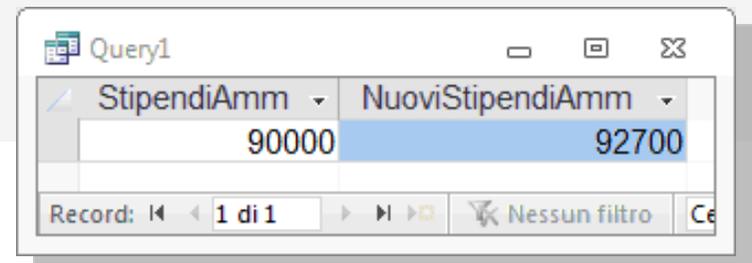


Query1

Expr1000	ResidentiRoma
2	2

Record: 1 di 1 | Nessun filtro

```
SELECT SUM(Stipendio) AS StipendiAmm,  
       SUM(Stipendio*1.03) AS NuoviStipendiAmm  
FROM Impiegati  
WHERE Dipartimento = 'Amm';
```



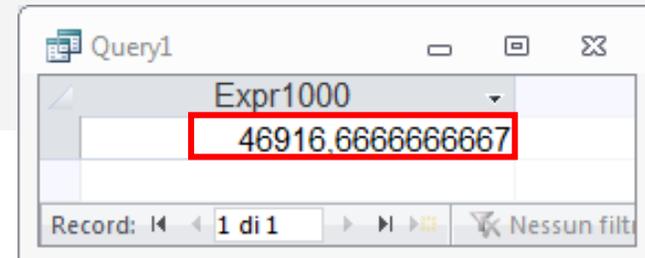
Query1

StipendiAmm	NuoviStipendiAmm
90000	92700

Record: 1 di 1 | Nessun filtro

# Funzioni di aggregazione (3)

```
SELECT AVG(Stipendio)
FROM Impiegati INNER JOIN, Dipartimenti
     ON Dipartimento = Codice
WHERE Sede = 'Torino';
```



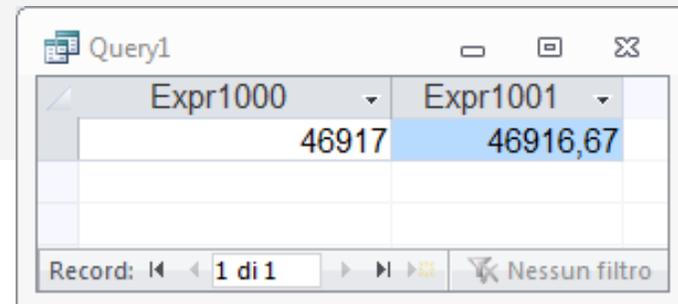
Query1

Expr1000
46916.6666666667

Record: 1 di 1

Nessun filtro

```
SELECT ROUND(AVG(Stipendio)), ROUND(AVG(Stipendio), 2)
FROM Impiegati INNER JOIN, Dipartimenti
     ON Dipartimento = Codice
WHERE Sede = 'Torino';
```



Query1

Expr1000	Expr1001
46917	46916,67

Record: 1 di 1

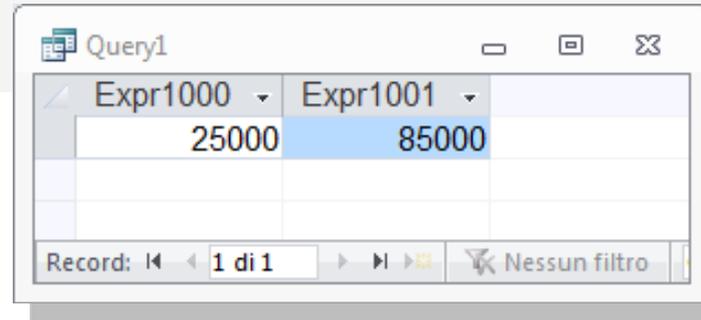
Nessun filtro

- Per arrotondare i risultati si usa la funzione **ROUND**

**ROUND( *Espressione*, *NumCifre* )**

# Funzioni di aggregazione (4)

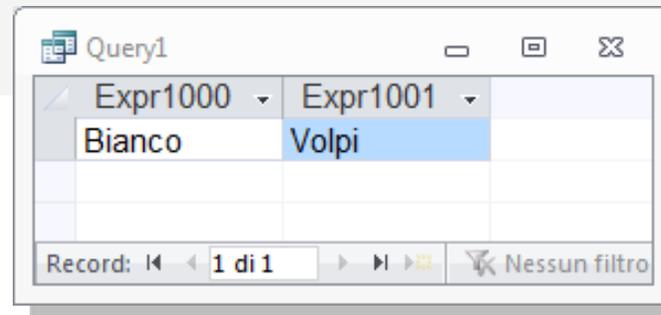
```
SELECT MIN(Stipendio), MAX(Stipendio)  
FROM Impiegati;
```



A screenshot of a query result window titled "Query1". The window displays a table with two columns, "Expr1000" and "Expr1001". The first row contains the values "25000" and "85000" respectively. The second row is empty. The status bar at the bottom indicates "Record: 1 di 1" and "Nessun filtro".

Expr1000	Expr1001
25000	85000

```
SELECT MIN(Cognome), MAX(Cognome)  
FROM Impiegati;
```



A screenshot of a query result window titled "Query1". The window displays a table with two columns, "Expr1000" and "Expr1001". The first row contains the values "Bianco" and "Volpi" respectively. The second row is empty. The status bar at the bottom indicates "Record: 1 di 1" and "Nessun filtro".

Expr1000	Expr1001
Bianco	Volpi

---

# Ordinamenti e Raggruppamenti

# Ordinamenti (1)

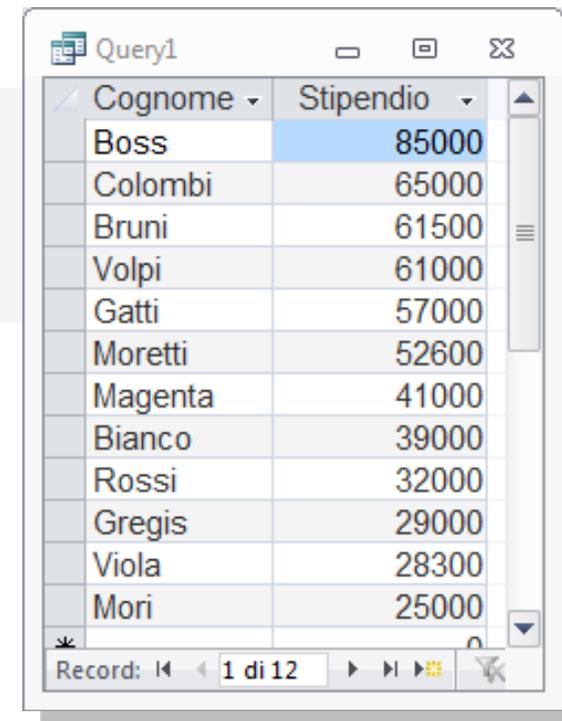
**ORDER BY**, se presente, deve essere l'ultima clausola di **SELECT**

```
SELECT Cognome, Nome, Residenza  
FROM Impiegati  
ORDER BY Cognome, Nome;
```

Ordinato per valori crescenti di Cognome e, a parità di Cognome, per Nome

```
SELECT Cognome, Stipendio  
FROM Impiegati  
ORDER BY Stipendio DESC, Cognome;
```

**ORDER BY** *Colonna* ASC | DESC



Cognome	Stipendio
Boss	85000
Colombi	65000
Bruni	61500
Volpi	61000
Gatti	57000
Moretti	52600
Magenta	41000
Bianco	39000
Rossi	32000
Gregis	29000
Viola	28300
Mori	25000

# Ordinamenti (2)

- Ordinare secondo i valori di un campo calcolato

```
1      2      3      4
SELECT NomeProdotto, PrezUnit, Qta, PrezUnit*Qta
FROM Fatture
ORDER BY 4 DESC;
```

Forma sconsigliata

```
SELECT NomeProdotto, PrezUnit, Qta, PrezUnit*Qta AS Totale
FROM Fatture
ORDER BY Totale DESC;
```

Forma raccomandata

- **Attenzione:** con Access bisogna usare una delle due forme

```
ORDER BY [PrezUnit]*[Qta] DESC
```

```
ORDER BY 4 DESC
```

# Raggruppamenti (1)

- **GROUP BY** per sintetizzare i valori di un campo per classi omogenee

```
SELECT Dipartimento, COUNT(ID) AS .. , SUM(Stipendio) AS ..  
FROM Impiegati  
GROUP BY Dipartimento;
```

Nella clausola **SELECT**  
possono comparire solo i campi  
elencati in **GROUP BY** e  
funzioni di aggregazione

Le righe sono raggruppate  
per dipartimento

Dipartimento	ID	Stipendio
	5	28300
Amm	12	61000
Amm	16	29000
Direz	13	85000
Direz	19	39000
Mag	6	25000
Mag	18	61500
Mkt	17	52600
Prod	1	32000
Prod	10	65000
Prod	11	41000
R&S	14	57000

Le funzioni di aggregazione sono  
applicate ai raggruppamenti

Dipartimento	Dipendenti	Stipendi
	1	28300
Amm	2	90000
Direz	2	124000
Mag	2	86500
Mkt	1	52600
Prod	3	138000
R&S	1	57000



# Raggruppamenti (2)

- La precedente interrogazione in **SQL** e **QBE** di Access

The screenshot displays the Microsoft Access interface for a query named 'Query2'. On the left, a field list for the 'Impiegati' table is shown, including fields: ID (primary key), Nome, Cognome, Residenza, Stipendio, and Dipartimento. The main window contains the following SQL code:

```
SELECT Impiegati.Dipartimento,  
       Count(Impiegati.ID) AS Dipendenti,  
       Sum(Impiegati.Stipendio) AS Stipendi  
FROM Impiegati  
GROUP BY Impiegati.Dipartimento;
```

Below the SQL editor, the QBE (Query By Example) grid is visible, showing the following structure:

Campo:	Dipartimento	Dipendenti: ID	Stipendi: Stipendio	
Tabella:	Impiegati	Impiegati	Impiegati	
Formula:	Raggruppamento	Conteggio	Somma	
Ordinamento:				
Mostra:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Criteri:				
Oppure:				

# Raggruppamenti (3)

- Volendo il **nome** del dipartimento, si deve raggruppare per *Descrizione*:

```
SELECT Descrizione, COUNT(*), SUM(Stipendio)
FROM Impiegati INNER JOIN Dipartimenti
ON Impiegati.Dipartimento = Dipartimenti.Codice
GROUP BY Descrizione;
```

- Oppure

```
SELECT Descrizione, COUNT(*), SUM(Stipendio)
FROM Impiegati INNER JOIN Dipartimenti
ON Impiegati.Dipartimento = Dipartimenti.Codice
GROUP BY Dipartimento, Descrizione;
```



# Condizioni sui raggruppamenti (1)

- **HAVING** per elencare i dipartimenti con più di due dipendenti

```
SELECT Dipartimento, COUNT(ID), SUM(Stipendio)
FROM Impiegati
GROUP BY Dipartimento
HAVING COUNT(ID) > 2;
```

The screenshot shows the Microsoft Access interface. At the top, a window titled 'Query2' displays a table view with the following data:

Dipartimer	ConteggioDiE	SommaDiStipe
Prod	3	138000

Below this, a larger window also titled 'Query2' shows the 'Impiegati' table structure and a field list for the query. The field list includes:

Campo:	Dipartimento	ID	Stipendio	[ID]
Tabella:	Impiegati	Impiegati	Impiegati	Impiegati
Formula:	Raggruppamento	Conteggio	Somma	Conteggio
Ordinamento:				
Mostra:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Criteri:				<input type="checkbox"/>
Operazioni:				<input type="checkbox"/>

A yellow callout box with the text 'HAVING predica sui raggruppamenti' has a dashed red line pointing to the 'HAVING COUNT(ID) > 2;' clause in the SQL code above and the '>2' condition in the field list below. The '>2' condition is also enclosed in a red box.

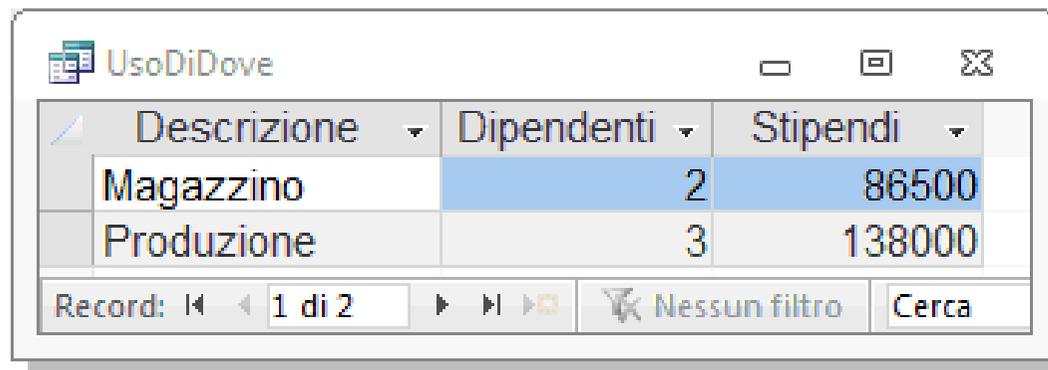
# Condizioni sui raggruppamenti (2)

- **HAVING e WHERE:** i dipartimenti di Torino con più di 1 dipendente

```
SELECT Descrizione, Count(ID) AS Dipendenti,  
        Sum(Stipendio) AS Stipendi  
FROM Dipartimenti D INNER JOIN Impiegati I ON  
        D.Codice = I.Dipartimento  
WHERE Sede = 'Torino'  
GROUP BY Descrizione  
HAVING Count(ID)>1;
```

**WHERE** predica  
sulle righe

**HAVING** predica sui  
raggruppamenti



Descrizione	Dipendenti	Stipendi
Magazzino	2	86500
Produzione	3	138000

# Condizioni sui raggruppamenti (3)

- **DOVE** nelle query QBE per inserire condizioni sulle righe

The screenshot shows the 'UsoDiDove' database tool interface. At the top, there are two tables: 'Dipartimenti' and 'Impiegati'. 'Dipartimenti' has fields: Codice (primary key), Descrizione, Sede, and Manager. 'Impiegati' has fields: ID (primary key), Nome, Cognome, Residenza, Stipendio, and Dipartimento. A 1-to-many relationship is shown between 'Dipartimenti' and 'Impiegati'. To the right, a SQL query window displays the following code:

```
...  
WHERE Sede = 'Torino'  
GROUP BY Descrizione  
HAVING Count(ID) > 1
```

Below the tables is a query design grid. The grid has columns for 'Descrizione', 'Dipendenti: ID', 'Stipendi: Stipendio', and 'Sede'. The 'Sede' column is highlighted with a red arrow. The 'Criteria' row shows a condition '>1' in the 'Dipendenti: ID' column and ''Torino'' in the 'Sede' column. A dropdown menu is open for the 'Sede' field, showing options like 'Raggruppamento', 'Somma', 'Media', 'Min', 'Max', 'Conteggio', 'DevSt', 'Var', 'Primo', 'Ultimo', 'Espressione', and 'Dove'. The 'Dove' option is highlighted with a red box. Two yellow callout boxes are present: one pointing to the '>1' condition with the text 'Condizione sui raggruppamenti', and another pointing to the ''Torino'' condition with the text 'Condizione sulle righe'.

Campo:	Descrizione	Dipendenti: ID	Stipendi: Stipendio	Sede
Tabella:	Dipartimenti	Impiegati	Impiegati	Dipartimenti
Formula:	Raggruppamento	Conteggio	Somma	Dove
Ordinamento:				
Mostra:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Criteria:		>1		'Torino'

# Il comando SELECT

- Tutte le clausole del comando **SELECT**

**SELECT** *Elenco di espressioni da mostrare*

**FROM** *Tabelle da cui estrarre le righe*

**WHERE** *Condizioni sulle congiunzioni e sulle righe estratte*

**GROUP BY** *Campi da considerare per i raggruppamenti*

**HAVING** *Condizioni sui raggruppamenti*

**ORDER BY** *Ordinamenti sulle espressioni elencate in SELECT*

- Obbligatorie le sole clausole **SELECT** (e **FROM** se ci sono tabelle)
- Devono rispettare l'ordinamento: **SELECT → FROM → ... → ORDER BY**
- **SELECT** è valutata dal motore del DBMS nell'ordine:  
**FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY**

---

## **Interrogazioni nidificate**

# Query nidificate (1)

- *Nome, Cognome e Dipartimento* del dipendente con lo stipendio massimo

Query1

```
SELECT MAX(Stipendio)
FROM Impiegati;
```

Expr1000

85000

Record: 1 di 1

Nota lo stipendio massimo ..

Query2

```
SELECT Nome, Cognome, Dipartimento
FROM Impiegati
WHERE Stipendio = 85000;
```

Nome	Cognome	Dipartimento
Ugo	Boss	Direz

Record: 1 di 1

Nessun filtro

- Perché non cortocircuitare le due interrogazioni?

# Query nidificate (2)

Una costante in una clausola **WHERE** può essere rimpiazzata con l'interrogazione che genera tale costante

```
SELECT Nome, Cognome, Dipartimento
FROM Impiegati
WHERE Stipendio = ( SELECT MAX(Stipendio)
                   FROM Impiegati );
```

**Sottointerrogazione:** deve essere racchiusa da parentesi.  
L'interrogazione è una sola e c'è un solo ; finale.

# Query nidificate (3)

- *Cognome, Nome e Stipendio* degli impiegati con stipendio inferiore alla media degli stipendi degli altri impiegati

```
SELECT Cognome, Nome, Stipendio
FROM Impiegati
WHERE Stipendio < ( SELECT AVG(Stipendio)
                    FROM Impiegati );
```

Cognome	Nome	Stipendio
Rossi	Mario	32000
Viola	Marco	28300
Mori	Enrico	25000
Magenta	Fabrizio	41000
Gregis	Elisabetta	29000
Bianco	Anita	39000

- Il dipartimento che spende di più in stipendi (problema già risolto)

```
SELECT Dipartimento, Stipendi
FROM StipendiPerDipartimento
WHERE Stipendi = ( SELECT MAX(Stipendi)
                  FROM StipendiPerDipartimento );
```

Quarto modo

# Query nidificate (4)

- Sottointerrogazioni che restituiscono un **elenco** di valori: *Nome*, *Cognome* e *Stipendio* dei dipendenti che sono manager.

**ElencoManager**

```
SELECT DISTINCT Manager  
FROM Dipartimenti;
```

Restituisce l'elenco di valori: 10,12,13,14

**DatiManager**

```
SELECT Nome, Cognome, Stipendio  
FROM Impiegati  
WHERE ID IN (10,12,13,14);
```

Perché non cortocircuitare le due query?

L'elenco di valori che compare nel predicato **IN**, in una clausola **WHERE**, può essere sostituito dalla sottointerrogazione che lo genera.

# Query nidificate (5)

SELECT Nome, Cognome, Stipendio  
FROM Impiegati  
WHERE ID IN ( SELECT DISTINCT Manager  
FROM Dipartimenti );

Nome	Cognome	Stipendio
Margherita	Colombi	65000
Franco	Volpi	61000
Ugo	Boss	85000
Mario	Gatti	57000

Bisogna usare un comando SELECT anche nelle query nidificate in modalità QBE

Campo:	Nome	Cognome	Stipendio	[ID]
Tabella:	Impiegati	Impiegati	Impiegati	Impiegati
Ordinamento:				
Mostra:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Criteri:				In (SELECT DISTINCT Manager FROM Dipartimenti )

# Numero di valori diversi

- Da quante differenti città di residenza provengono i dipendenti del dipartimento *Produzione*?

```
SELECT COUNT(DISTINCT Residenza)
FROM Impiegati
WHERE Dipartimento = 'Prod';
```

Sintassi non ammessa  
nell'SQL di Access

- Bisogna ricorrere a una query annidata nella clausola **FROM**

```
SELECT COUNT(*)
FROM ( SELECT DISTINCT Residenza
      FROM Impiegati
      WHERE Dipartimento = 'Prod');
```

# Ricerca di valori duplicati

- Chi sono i manager responsabili di più di un dipartimento?

Query di ricerca duplicati di Access

Manager	Descrizione
10	Produzione
10	Magazzino
12	Personale
12	Amministrazione
13	Marketing
13	Direzione Generale
*	0

```
SELECT Manager, Descrizione
FROM Dipartimenti
WHERE Manager IN ( SELECT Manager FROM Dipartimenti
                   GROUP BY Manager
                   HAVING Count(*) > 1 )
ORDER BY Manager;
```